# Transitional function Rationale

Huy Nguyen

February 2016

## 1 Introduction:

In order to build sets of ancestral gene/genes blocks, I try to traverse the phylogenetic tree from leaf to the root by following methods:

1. Consider each non-leaf node at each level (bottom-up tree traversal).
2. All non-leaf nodes a given level are considered before moving on to the next higher level.

To build ancestral gene blocks x from given children A and B, I construct set of gene / gene blocks that x has to include. Define this set as S(x). For each internal node x, define set Leaf(x)=$\{$leaf node A $\|$ A can reach x using the above traversal method$\}$

I categorize internal nodes of the tree into 3 different kind:

1. The one that its 2 children are leaf nodes. Define these internal nodes as double leaf (dl)
2. The one that has 1 of its children as a leaf nodes. Define these internal nodes as half leaf (hl)
3. The one that none of its children is leaf node. Define these internal nodes as no leaf (nl)

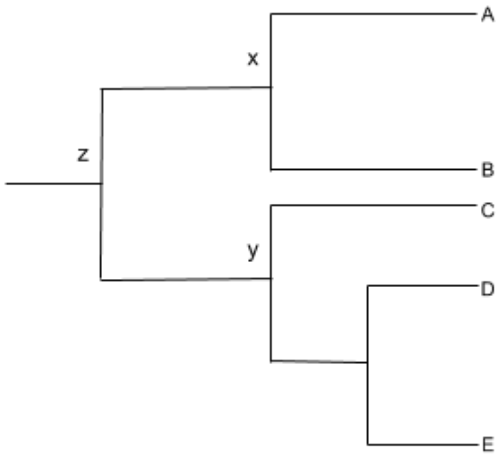


Figure 1: a tree example: x is dl, y is hl, z is nl

## 2 Data Structure:

The data structure for each internal nodes is a tupple that contains following information:

1. Set(x) at index 0.
2. Dictionary that has:
   - key: gene g.
   - value: list of length 2 that contains
     - index 0: count of gene g in x.

– index 1: number of leaf nodes in Leaf(x) that contains gene g.

3. Size of Leaf(x)

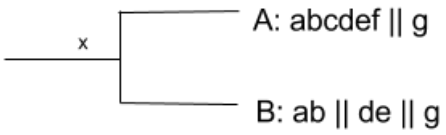


Figure 2: x is a tuple ($\{$ab,de,g$\}$,$\{$a:[2,2],b:[2,2],c:[1,1],d:[2:2],e:[2,2],f:[1,1:],g[2:2]$\}$,2)

Because of this, to decide the ancestral set of genes/ blocks of genes, I provide 3 functions with different parameters:

1. GenomevsGenome(Genome1,Genome2)
2. SetvsGenome(myTuple,Genome)
3. SetvsSet(myTuple1,myTuple2)

# 3 Goal:

The main point here is to determine whether I should include a gene g in the higher level internal node. In order to do this, I need to have a count of gene g from each of its children, and build a transitional function. In this paper, I will present my transitional function, and rationale why I choose it (hopefully with solid proof).

# 4 Assumption:

For any children A,B and its closest common ancestor x. Consider gene g in our reference operon:

1. Each leaf node has count(g) of value either 0 or 1.
2. Each internal node has count(g) of value either 0, 1, or 2.
3. S(x) includes g if and only if count(g) in x is 2.
   - If count(g) = 0, then apparently S(x) should not include g
   - If count(g) = 1, either include g in S(x) or not does not change
4. count(g) in x depends on the count(g) in A and B. However, in some special cases, it also depends on the frequency it appears in all Leaf(x). Define $FREQ_g(x)$ as frequency of gene g in Leaf(x). There are 3 scenarios:
   (a) if frequency is less than .5 then count(g) is 0
   (b) if frequency is greater or equal to .5, but less than .67 (2/3), then count(g) is 1
   (c) if frequency is greater or equal to .67, then count(g) is 2

# 5 Transitional Function:

## 5.1 GenomevsGenome:

Given children Genome A and Genome B, define their closest common ancestor as x. Consider gene g in our reference operon:

- $A.count(g) = 0, B.count(g) = 0 \rightarrow x.count(g) = 0.$
- $A.count(g) = 0, B.count(g) = 1 \rightarrow x.count(g) = 1.$
- $A.count(g) = 1, B.count(g) = 0 \rightarrow x.count(g) = 1.$
- $A.count(g) = 1, B.count(g) = 1 \rightarrow x.count(g) = 2.$

## 5.2 SetvsGenome:

Given children set A and genome B, define their closest common ancestor as x. Consider gene g in our reference operon, define FREQ is the frequency of gene g in the set leaf(x).

- $A.count(g) = 0, B.count(g) = 0 \rightarrow x.count(g) = 0.$
- $A.count(g) = 0, B.count(g) = 1 \rightarrow x.count(g) = \begin{cases} 0, & \text{if } FREQ_g(x) < .25 \\ 1, & \text{if } .25 \leq FREQ_g(x) < .5 \end{cases}$
- $A.count(g) = 1, B.count(g) = 0 \rightarrow x.count(g) = \begin{cases} 0, & \text{if } FREQ_g(x) < .25 \\ 1, & \text{if } .25 \leq FREQ_g(x) < .5 \end{cases}$
- $A.count(g) = 1, B.count(g) = 1 \rightarrow x.count(g) = \begin{cases} 1, & \text{if } .25 \leq FREQ_g(x) < .5 \\ 2, & \text{if } .5 \leq FREQ_g(x) \end{cases}$
- $A.count(g) = 2, B.count(g) = 0 \rightarrow x.count(g) = \begin{cases} 1, & \text{if } .25 \leq FREQ_g(x) < .5 \\ 2, & \text{if } .5 \leq FREQ_g(x) \end{cases}$
- $A.count(g) = 2, B.count(g) = 1 \rightarrow x.count(g) = 2.$

## 5.3 SetvsSet:

Given children set A and set B, define their closest common ancestor as x. Consider gene g in our reference operon, define FREQ is the frequency of gene g in the set leaf(x). Because the matrix will be symmetrical, hence, we only consider the one half of it

- $A.count(g) = 0, B.count(g) = 0 \rightarrow x.count(g) = 0.$
- $A.count(g) = 0, B.count(g) = 1 \rightarrow x.count(g) = \begin{cases} 0, & \text{if } FREQ_g(x) < .25 \\ 1, & \text{if } .25 \leq FREQ_g(x) < .5 \end{cases}$
- $A.count(g) = 0, B.count(g) = 2 \rightarrow x.count(g) = \begin{cases} 0, & \text{if } FREQ_g(x) < .25 \\ 1, & \text{if } .25 \leq FREQ_g(x) < .5 \\ 2, & \text{if } .5 \leq FREQ_g(x) \end{cases}$
- $A.count(g) = 1, B.count(g) = 1 \rightarrow x.count(g) = \begin{cases} 1, & \text{if } .25 \leq FREQ_g(x) < .5 \\ 2, & \text{if } .5 \leq FREQ_g(x) \end{cases}$
- $A.count(g) = 1, B.count(g) = 2 \rightarrow x.count(g) = \begin{cases} 1, & \text{if } .25 \leq FREQ_g(x) < .5 \\ 2, & \text{if } .5 \leq FREQ_g(x) \end{cases}$
- $A.count(g) = 2, B.count(g) = 2 \rightarrow x.count(g) = 2.$

## 5.4 Property of the result

Here are the list of properties of our ancestral gene/ gene blocks x from any children A, B:

1. For any genes g include in S(x), $FREQ_g(x) \geq .5$
2. For any genes g that does not appear in S(x), $FREQ_g(x) < .5$